基于深度学习的无碰撞引力N体数值模拟的可行性研究*

赵梓成[1,2]，龙潜[1]，董小波[1]，孟润宇[1,2]，钟诗言[1]，谌俊毅[1]，向梓琨[3]
(1. 中国科学院云南天文台，云南 昆明 650216；2.中国科学院大学，北京 100049；3.中国科学技术大学 物理学院，安徽 合肥 230026)

摘 要： 本文提出，用深度神经网络代替快速傅里叶变换法，求解无碰撞引力N体数值模拟方法PM-Tree中的势能，以提升PM-Tree方法的效率，验证深度学习方法加速无碰撞引力N体数值模拟的可行性。无碰撞引力N体数值模拟对研究星系、暗物质晕以及宇宙大尺度结构的形成和演化都有重要意义。而无碰撞引力N体数值模拟的传统方法在大规模问题上的模拟计算非常耗时，其中常用的PM-Tree方法的主要耗时部分是求解势能（解Poisson方程）。本文提出使用深度神经网络代替传统方法加速求解Poisson方程，多次调整并训练和测试深度神经网络模型结构，最终选用了辅以残差网络局部结构的Encoder-Decoder整体结构。验证了深度神经网络解Poisson方程的计算时间复杂度为$O(N)$；同样数据下进行测试，速度快于快速傅里叶变换方法求解和有限差分法求解；在同等采样率的情况下，精度优于快速傅里叶变换方法求解；并且具有可扩展性。故无碰撞引力N体数值模拟中，用深度神经网络可以提升PM-Tree方法中求解势能的速度，从而有效提升整体模拟速度。

关键词： 泊松方程；引力N体数值模拟；深度神经网络；
中图分类号： P132+.4;TP183　　**文献标识码：** A　　**文章编号：**
1

引力N体模拟是研究星系、暗物质晕以及宇宙大尺度结构等天文系统的形成和演化的主要方法之一。由于引力是长程力，对于星系以及更大尺度或更多粒子数的引力系统，可以把其成员恒星（或暗物质粒子）的运动看作是在总体的引力势下的运动，而不需要考虑恒星之间（或暗物质粒子之间）的相互作用（即“碰撞”），此即无碰撞引力系统的概念，也即是一种平均场近似的思想。对这类系统随时间演化的N体数值模拟，可不考虑“N体”粒子两两之间的引力作用，而仅需计算共同的引力场（即解Poisson方程），以及该引力场对具体粒子的作用，此即无碰撞引力N体数值模拟[1]。在近来的应用中（Gadget-3,Gadget-4），PM-Tree方法是无碰撞引力N体模拟的主要方法之一，而PM-Tree方法中求解势能（解Poisson方程）的步骤是主要耗时的部分[2]。所以求解Poisson方程的速度是影响无碰撞引力N体数值模拟耗时的主要因素之一。

在具体的引力N体模拟中，“N体”粒子的数目N不一定对应，往往不可能达到恒星或暗物质粒子的真实数量；数目N的设定，取决于模拟精度的需求以及计算能力。传统无碰撞引力N体数值模拟方法的计算速度，现在适用于模拟粒子数规模在10^6-10^{13}之间的宇宙系统[1]。模拟更大规模的宇宙系统的分布演化则需要计算速度更快的方法。而深度神经网络在诸多解方程问题中都被证明有效，并且还有速度快的特点，所以我们提出，在传统无碰撞引力N体数值模拟方法PM-Tree中，使用深度神经网络代替快速傅里变换法计算主要耗时的求解势能部分（解Poisson方程）。以粒子数规模为10^{13}的系统为例，用深度神经网络的方法，单次势能求解的速度理论上能快40倍左右，循环叠加以后对整体模拟的加速较为可观。并且从各方面说明了深度神经网络加速求解Poisson方程的有效性，为后续加速无碰撞引力N体数值模拟的研究打下基础。而且因为深度神经网络具有可扩展性的特点，所以在其他需要求解Poisson方程的物理问题中，我们提出的深度神经网络模型也能适用。

引力N体数值模拟方法：其一，Particle-to-Particle(PP)方法，直接计算粒子间的牛顿引

---

*基金项目：国家自然科学基金资助项目(11773074,11873083)资助.
收稿日期：2020-12-02；　修订日期：2021-4-9
作者简介：赵梓成，男，硕士研究生.研究方向：引力N体数值模拟和深度学习.Email:zhaozicheng@ynao.ac.cn
通讯作者：龙潜，男，研究员.研究方向：天文信息技术和人工智能.Email:longqian@ynao.ac.cn

力作用，计算时间复杂度为$O(N^2)$[3]。其二，当系统中粒子数量较多时，可以使用自1980年代延用至今的Tree代码加快模拟速度[4-5]。Tree代码使用递归方式计算子树，所以省去了重复计算，将引力N体数值模拟的计算时间复杂度降低到了$O(Nlog(N))$，例如有GASOLINE[6]，iVINE[7]等应用。其三，有Particle Mesh(PM)方法：首先使用Cloud-In-Cell(CIC)方法将粒子的位置信息转换为均匀网格的密度信息，然后通过快速傅里叶变换（FFT）方法来求解基于这些网格的Poisson方程。该方法的计算时间复杂度为$O(Nlog(N))$[8]，N代表网格单元数。其四，为了提高PM方法中网格内模拟粒子的精确率，在网格内使用Tree代码来计算粒子间的相互作用，称为PM-Tree方法[9]。由于Tree代码每次在PM方法的一个小步骤内并行计算，而且一次Tree代码计算时间小于或等于PM方法中一个小步骤的二分之一计算时间[10]，所以PM-Tree方法的计算时间复杂度和PM方法相同，为$O(Nlog(N))$。至今，PM-Tree方法为无碰撞引力N体数值模拟的主要方法之一，例如有Gadget-2[11]等应用。最后，近年来的N体程序（如Gadget-3,Gadget-4）[2]还实现了快速多极子方法（Fast Multipole Method；FMM）[12]。FMM可以将计算时间复杂度降到$O(N)$，在相同精度下可以优于Tree方法的性能表现，但FFM算法更为复杂，特别是在动态步长、特殊边界条件、高效并行化等方面实现起来较为困难。

以上传统引力N体数值模拟方法的计算速度较慢，在超大规模粒子数量问题中难以适用，所以需要速度更快的方法。目前深度神经网络在很多领域的成果显著[13-14]，并且神经网络也常被用于解方程[15-16]。Lagaris等证明了神经网络能解常微分方程和偏微分方程[17]。引力N体数值模拟问题中的高维数据所需计算量巨大，而卷积神经网络的核心能力就是从高维输入中提取多尺度特征[18-20]，这能有效应对数值解偏微分方程中的维度诅咒问题[21]。所以我们提出采用深度卷积神经网络的方法，在无碰撞引力N体数值模拟方法PM-Tree中，代替快速傅里叶变换法，更快求解势能（解Poisson方程）。

1　相关工作

近几年出现了一些相关使用深度神经网络求解偏微分方程的方法。由无碰撞玻尔兹曼方程的速度矩定义的Jeans方程，与流体力学中（无粘的）Navier-stokes方程具有相同的形式，所以在这类流体领域解Navier-stokes方程的相关研究结论也适用于引力N体数值模拟[22]。稳定流近似是一个Navier Stokes流体问题,是一种难以计算且非常耗时的迭代过程，Guo使用卷积神经网络实现了快速准确的拟合[23]。Zhu等在随机多孔介质流动问题中使用卷积神经网络，仅用很少的数据就得到了很好的效果，优于蒙特卡洛法[24]。Saakaar等证明了在流场预测问题上使用卷积神经网络方法的有效性，而且速度比Reynolds Averaged Navier-stokes法快[25]。Khoo等证明了神经网络可以简洁地解具有不确定性的偏微分方程问题[26]。Adler等在ill-posed逆问题上证明使用深度神经网络迭代求解，速度提升十分显著[27]。

不过这类用深度神经网络求解偏微分方程的方法是将求解算子参数化，并不直接，而且对于网格化尺寸不具有鲁棒性，但对于引力N体粒子数值模拟来说，将要处理的问题是不停迭代的，需要更简单高效的求解方法。而最近出现了一些针对特定问题直接求解偏微分方程的方法。Wei Tang等在电磁领域求解二维图像形式的Poisson方程，得到了很好的效果[28]。Weinan等提出了Deep Ritz方法，在使用Ritz方法求解偏微分方程过程中使用深度神经网络来寻找Ritz法的解，证明了深度神经网络的方法优于有限差分法[29]。Kaushik等在无限维空间中证明了用深度神经网络参数化求解椭圆偏微分方程问题的有效性和收敛性[30]。Smith等创作了一个EikoNet,证明深度神经网络的方法能够稳定求解Eikonal方程，并能应用到各种领域，且具有非常显著的计算速度优势[31]。

2　本文工作及贡献

我们结合目前深度学习的特点：速度快、有效解决高维度问题、能求解偏微分方程等，提出使用深度神经网络代替快速傅里叶变换方法，求解无碰撞引力N体数值模拟方法PM-Tree中的

势能（解Poisson方程）。这种方法理论上可以更快地完成大规模的无碰撞引力N体数值模拟。并且它的可扩展性好，易于实现。我们搭建好的深度神经网络模型，以及对应的训练和测试代码都在Github中开源，项目地址：
https://github.com/hi76/Poisson-slover-based-on-Encoder-Decoder-Neural-Network

3 方法

N体数值模拟最早的直接求解法耗时巨大。平均场近似思想将难解的N体数值模拟转化为一个近似的统计问题，更易求解，但速度依然不快，后来陆续出现了一些优化的方法，如PM方法、PM-Tree方法、FMM方法等。其中PM-Tree方法相比于其他大多数N体数值模拟方法，有速度快、精度高的优点，所以是目前主流的方法之一。在PM-Tree方法中主要耗时的步骤是求解势能（解Poisson方程）。我们提出用深度神经网络模型代替快速傅里叶变换法，加速PM-Tree方法中势能的求解，以验证深度学习方法在加速无碰撞引力N体数值模拟中的可行性。

3.1 N体数值模拟

在物理学和天文学中，在一些物理作用力的影响下，对一个由粒子组成的系统的动力学模拟称为N体模拟。引力N体模拟主要是对N粒子引力相互作用方程求数值解。引力N体模拟在天体物理学中是一个常用的工具，在少体系统到宇宙大尺度结构，比如地球-月球-太阳系统到银河系中都有应用。N粒子引力相互作用方程为[32]：

$$\vec{F}_i = -\sum_{j\neq i} G\frac{m_i m_j(\vec{r}_i-\vec{r}_j)}{\left|\vec{r}_i-\vec{r}_j\right|^3} - \vec{\nabla}\cdot\Phi_{ext}(\vec{r}_i)\,,\ i=1,\ldots,n\#(1)$$

这里$G = 6.67300\cdot10^{-11}|m^3|kg^{-1}|s^{-2}|$，$\Phi_{ext}$是外部势能。

N粒子引力相互作用系统定义了一个6N+1维的相空间，这个相空间中每个时刻每个粒子都由N个位置和速度矢量确定。这个方程的解则是相空间中的一个运动轨迹。直接求解这个方程十分耗时，所以需要简化的数值方法求解。

3.2 平均场法

平均场法是出现较早的一种N体数值模拟的方法。用统计方法来描述引力N体问题时，系统分布$f(x_1,v_1,\cdots,x_n,v_n,t)$描述了粒子的全部信息。当不考虑粒子间的关联，即认为$f^{(2)}(x_1,v_1,x_2,v_2,t) = f^{(1)}(x_1,v_1,t)f^{(1)}(x_2,v_2,t)$，此即平均场近似，即无碰撞N体系统。平均场近似中，粒子受到的作用力取决于分布函数，而不必计算粒子两两之间的引力相互作用，这时我们可用单粒子分布函数$f(x,v,t)$来构造系统的平均场描述，即在d^3xd^3v大小的6维相空间中，发现一个粒子的概率为$f(x,v,t)d^3xd^3v$。这个分布函数的演化可以由无碰撞玻尔兹曼方程来描述[1]：

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + \vec{v}\frac{\partial f}{\partial\vec{x}} - \frac{\partial\Phi_T}{\partial\vec{x}}\cdot\frac{\partial f}{\partial\vec{v}} = 0\#(2)$$

总潜在势能为外部势能以及内部势能的和：

$$\Phi_T = \Phi_{ext}(\vec{x},t) + \Phi(\vec{x},t)\#(3)$$

这个内部势能即所有N体粒子的引力势能所构成的共同的自引力势，可基于所有粒子的密度分布$\rho(\vec{r},t)$，通过求解Poisson方程得到：

$$\nabla^2\Phi(\vec{x},t) = 4\pi G\rho(\vec{r},t)\#(4)$$

这里$\rho(\vec{r},t) = \int f(\vec{x},\vec{v},t)d^3v$，即与分布函数相对应的该系统的密度分布。平均场法将难解的N体问题转换为一个近似的统计问题，使其更易求解。但它的直接计算比较耗时，所以后来陆续出现了很多优化的方法，如PM方法[8]等。

3.3 PM-Tree

PM-Tree方法也是一种N体数值模拟的方法。它在PM方法的网格内加入Tree代码以实现更精细的模拟。因其速度较快且精度较高的特点，在最近的N体数值模拟应用中，常以PM-Tree方法为主进行模拟[2]。PM-Tree方法在每个时间步长（取决于模拟的时间区间，以及模拟的精度要求）内模拟的基本步骤可以概括为[10]：

（1）首先确认Tree代码应用的网格以及当中的粒子，除此以外的粒子只由PM方法来模拟计

算。

(2) 运行PM方法的一半步骤。

(3) 计算每个Tree代码应用的网格对应的外部总势能——其余网格对它的势能和。

(4) 对这些应用的网格使用Tree代码同时计算，并将计算结果加入到正在运行的PM方法步骤中。

(5) 计算PM方法中的势能以及对应每个粒子的加速度。

(6) 将每个Tree整合到PM方法的最后步骤。

(7) 更新粒子速度和位置信息。再从（1）步骤开始循环计算。

因为在PM-Tree方法步骤中，Tree代码是和PM方法并行计算的，并且每次Tree代码计算所需时间不大于PM方法每次计算耗时的一半，所以PM-Tree方法耗费的时间主要为PM方法步骤耗费的时间。在PM方法中，求解势能是主要耗时的计算步骤，而势能需要求解Poisson方程得到。所以加速PM-Tree方法的关键点在于加速求解Poisson方程。PM-Tree方法的应用中，常用快速傅里叶变换法求解Poisson方程[33]。本文提出用深度神经网络代替快速傅里叶变换法求解PM-Tree方法中的所有Poisson方程，以加速PM-Tree方法，也即是加速N体数值模拟。

3.4 Poisson方程数值求解

传统求解Poisson方程的方法有：有限元法、有限差分法、有限体积法、快速傅里叶变换求解法等。利用计算机数值求解Poisson方程，首先要将Poisson方程转为代数方程，即差分方程。二维Poisson方程的差分方程表示为：

$$\frac{\phi_{i+1,j}+\phi_{i-1,j}+\phi_{i,j+1}+\phi_{i,j-1}-4\phi_{i,j}}{DT^2}=4\pi G\rho_{i,j}\#(5)$$

分子即是对ϕ做卷积，DT为离散化参数。公式（5）在计算机中的操作可以简化表示为，生成的势能ϕ和一个二阶差分算子∇的卷积：

$$4\pi G\rho=\frac{\begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix}}{DT^2}\otimes\phi\#(6)$$

⊗是卷积操作符，这里卷积操作符的左边项即是二阶差分算子∇。有限差分法求解Poisson方程，是将公式（5）全部展开为一个有解的线性方程组，再用各类迭代法求解。有限差分法求解Poisson方程的计算时间复杂度为$O(N^2)$[34]。

快速傅里叶变换方法（FFT），利用傅里叶变换，将实数域中的卷积转换为复数域的乘法，进而将公式（6）转换为一个复数域中易解的方程，求解后再做傅里叶逆变换，即得Poisson方程的数值解[35]。快速傅里叶变换求解方法的计算时间复杂度为$O(NlogN)$。因其速度较快，在近来的引力N体数值模拟应用（Gadget-3,Gadget-4等）中，涉及到求解Poisson方程的步骤，主要也是用快速傅里叶变换方法进行求解[2]。

三维Poisson方程的二阶差分算子∇为：

$$\nabla=\frac{\left[\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix},\begin{bmatrix}0&1&0\\1&-6&1\\0&1&0\end{bmatrix},\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}\right]}{DT^2}\#(7)$$

FFT求解三维Poisson方程的原理和过程与二维相同，只是二阶差分算子∇不同，计算量更大一些。我们的目标是训练一个深度神经网络，以$O(N)$的计算时间复杂度，实现求解公式（6）的反卷积过程，进而验证深度神经网络代替快速傅里叶变换方法求解Poisson方程，并加速无碰撞引力N体数值模拟的可行性。因为我们关注的问题为：深度神经网络是否可以实现求解公式(6)的反卷积过程。所以我们选用消耗计算资源更少的二维Poisson方程进行验证即可。

目前N体数值模拟的粒子数规模主要在10^6-10^{13}之间。随着计算机计算能力的提升，能够模拟的粒子数规模以每年两倍左右增长，符合摩尔定律。在无碰撞引力N体数值模拟中，影响模拟速度的主要因素为解Poisson方程的速度，传统方法中常用的FFT求解的计算时间复杂度为$O(Nlog(N))$[1]。而我们提出的深度神经网络模型的计算时间复杂度为$O(N)$，以粒子数规模为10^{13}

为例，每一次Poisson方程求解的速度理论上能快40倍左右。在PM-Tree方法中，求解Poisson方程的步骤需要循环执行很多次，所以循环叠加以后，计算时间复杂度$O(N)$的方法对无碰撞引力N体数值模拟的加速将较为可观。而且随着将来模拟的粒子数规模不断增大，计算时间复杂度$O(N)$的方法所能节省的模拟时间也会更加明显。

3.5 深度神经网络模型

深度神经网络具有非常强的高维特征提取能力。在深度神经网络内部，是一个非常高维的特征空间，所以一个复杂的深度神经网络模型，对于变化不是过于剧烈的输入数据，都具有鲁棒性。而二维数据变为三维数据，相对于深度神经网络内部非常高维的特征空间，不是巨大的变化。而且即使数据复杂度变化较大，只要特征原理不变，深度神经网络也可以相应地增加模型的复杂度，而维持原有的效果[18-20]。所以我们的主要目标是验证深度神经网络能够实现求解公式（6）的反卷积过程。我们前后实验了数十种网络结构用于快速求解Poisson方程，最终找到了目前效果最好的网络结构。因为在机器翻译中，将一种语义转化为另一种语义的端到端的神经网络结构采用较多的是Encoder-Decoder结构[36]，这与公式(4)中的$\phi(\vec{x},t)$与$\rho(\vec{r},t)$的转化有类似之处，所以我们采用Encoder-Decoder这样的总体架构。网络基础模块借鉴ResNet[37]结构，第一种将一个卷积层的输入与它的输出相加(残差卷积结构)，另一种将一个卷积转置层的输入与它的输出相加(残差反卷积结构)。为了保证输入n大小尺寸的模拟分布图ρ，能够得到同样大小尺寸的模拟势能图ϕ，我们采用了完全对称的网络结构，前半段使用一次卷积，一次残差卷积，连续三次。然后做通道数不变的卷积与反卷积,用于提取图像的高维特征。后半段使用一次残差反卷积，一次卷积，连续三次，用于将高维特征表达为需要的势能分布，如图1所示。其中所有的卷积层或者反卷积层的核大小为3、步长为1、填充大小为1。激活函数均采用ReLU函数外套一层线性函数。

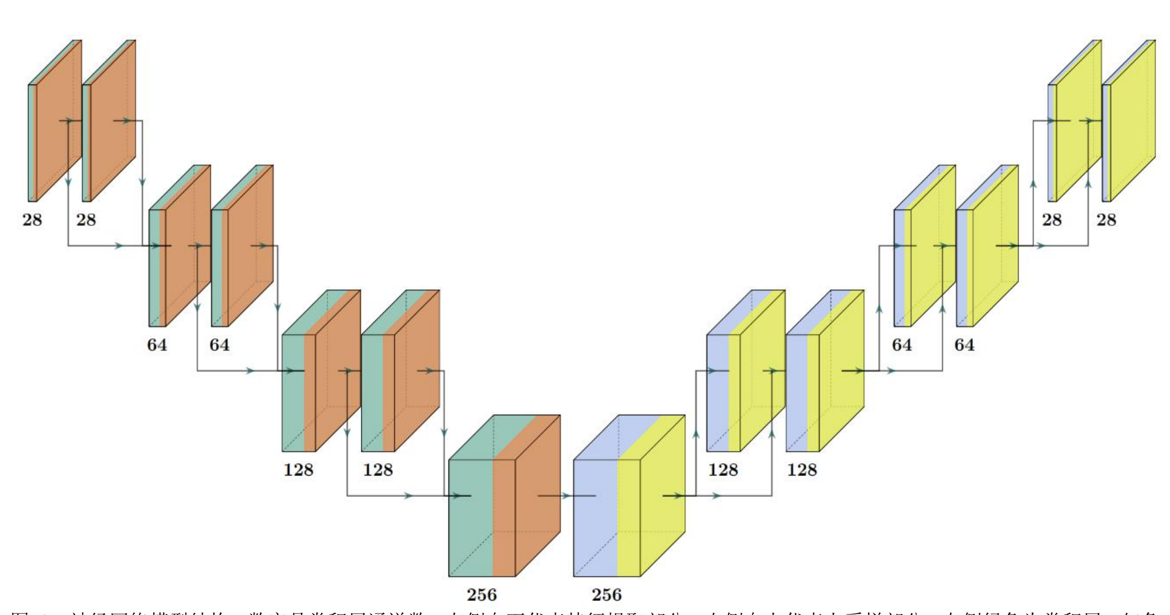


图 1　神经网络模型结构。数字是卷积层通道数，左侧向下代表特征提取部分，右侧向上代表上采样部分。左侧绿色为卷积层，红色为激活函数，右侧蓝色为反卷积层，黄色为激活函数。

Fig. 1　Neural network model Structure. The numbers are convolutional channels, with the left side down for feature extraction and the right side up for up-sampling. On the left side, green is the convolutional layer, red is the actication function, on the right side, blue is the convolutional transpose layer, yellow is the activation function.

我们的训练目标是最小化预测值和势能真值的平均平方误差:

$$MSE=\frac{1}{N}\sum(predict-\Phi)^2\#(8)$$

使用最大差值绝对值来评估误差:

$$error=Max|Predict-\Phi|\#(9)$$

卷积神经网络模型的计算时间复杂度为[38]：

$$O\left(\sum_{l=1}^{d} n_{l-1}\cdot s_l^2\cdot n_l\cdot m_l^2\right)\#(10)$$

这里l是卷积层索引，d是网络深度，n_l是第l层的卷积核数量，n_{l-1}是第l层的输入通道数。s_l是卷积核尺寸，m_l是输出特征图尺寸，N等于最后一层的输出特征图尺寸m_d^2。由于神经网络模型中间参数量固定，且通常无碰撞引力N体数值模拟中的$N \gg l,d,n_l,s_l,m_l(l \neq d)$，所以本文神经网络模型的计算时间复杂度为$O(N)$。本文实验也进一步证明了神经网络模型为线性复杂度的函数。

4　训练模型

我们的目标是：用深度神经网络代替快速傅里叶变换法，求解PM-Tree方法中的势能（解Poisson方程），以验证深度学习加速无碰撞引力N体数值模拟的可行性。其中关键任务是验证深度神经网络是否可以实现求解公式（6）中的反卷积过程，所以需要训练得到一个可以作为Poisson方程求解器的深度神经网络模型。使用随机生成的矩阵模拟网格化后需要预测的势能ϕ，即是Poisson方程的数值解。对生成图像做二阶差分得到模拟的输入分布ρ。建立一个深度神经网络模型，以多个卷积层来提取特征，多个反卷积层来表达特征输出预测势能。生成百万张数据图，用平均平方误差对这个深度神经网络模型进行训练。最后对训练好的模型做精度测试、用于简单等时势问题进行评估、并和快速傅里叶变换方法做精度对比、和快速傅里叶变换方法以及有限差分法做时间对比、用于多种网格化方式进行测试。

4.1　数据集

我们生成一百万张图作为训练集，一万张图作为测试集。这里我们假设将求解区域采样为$N \times N$的网格，则使用的数据中模拟势能ϕ真值的构成为：随机生成的$N \times N$矩阵，然后将周围一圈值设置为0作为Dirichlet边界条件。采用Dirichlet边界条件的原因是：在引力N体数值模拟中，通常我们所关注的物质团周围很大区域都是真空，可以近似为0。并且为了避免网络训练过于受到随机函数本身影响(随机函数得到的模式几乎都是不规律跳变值,相同的或平滑的情况很少)，我们也人为加入了一些特殊模式作为数据增强:对中间部分区域做高斯平滑、将中间部分区域数值设置为0，将四角部分区域数值设置为0.99等。这些图像是模拟的Poisson方程的解，也就是势能分布ϕ，是深度神经网络模型的预测目标，如图2-5所示。

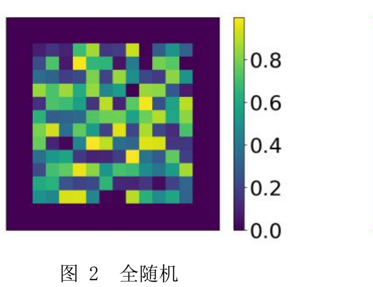


图 2　全随机
Fig.2　Rand

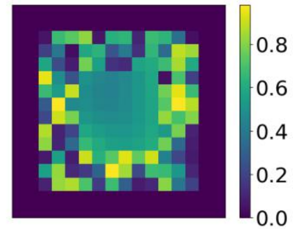


图 3 中间平滑
Fig.3　Middle Smooth

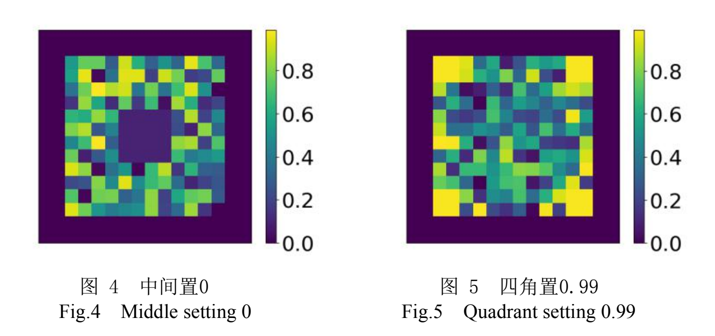


图 4　中间置0
Fig.4　Middle setting 0

图 5　四角置0.99
Fig.5　Quadrant setting 0.99

需要模拟求解的分布ρ图，可由以上模拟的势能ϕ通过公式（6）得到，如图6-9所示，这些分布ρ图也即是深度神经网络模型的输入。

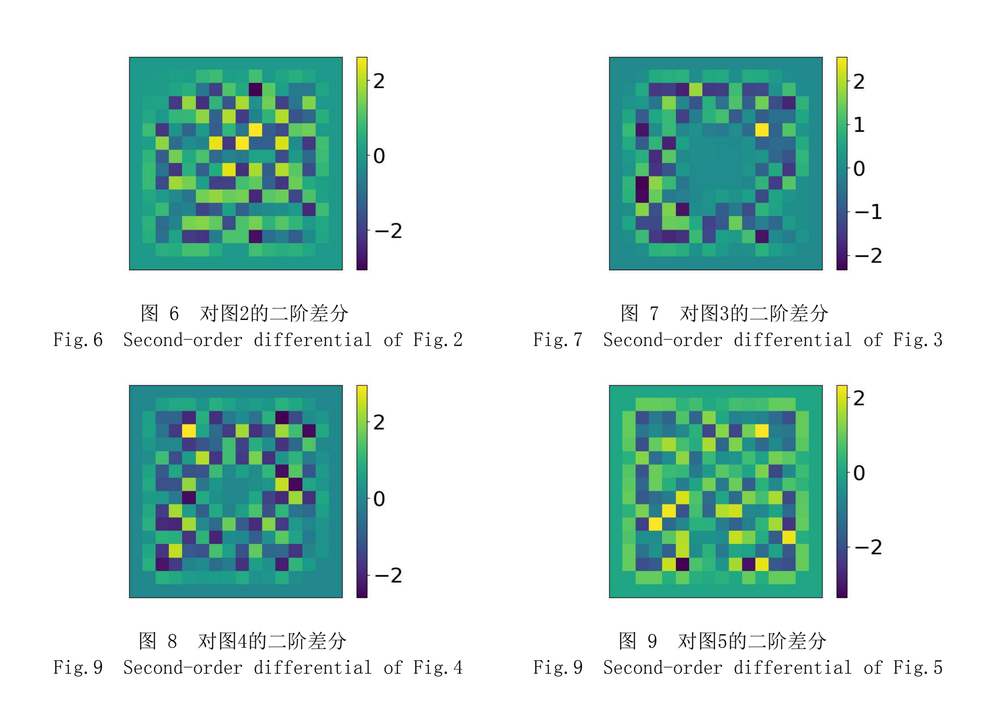


图 6　对图2的二阶差分
Fig.6　Second-order differential of Fig.2

图 7　对图3的二阶差分
Fig.7　Second-order differential of Fig.3

图 8　对图4的二阶差分
Fig.9　Second-order differential of Fig.4

图 9　对图5的二阶差分
Fig.9　Second-order differential of Fig.5

每次训练给深度神经网络模型输入ρ，用模型输出与ϕ的平均平方误差进行训练。因为我们的数据是随机生成的，所以有不限数量的训练数据的优点。每一次生成一百万对数据用于训练。

4.2 训练参数

我们在Pytorch中实现深度神经网络模型。每一次训练使用一百万对数据，批大小设置为100。优化器选用Adam[39]，学习率从0.01视情况调整到0.000001。每一组数据训练100次。训练使用的GPU是Tesla P100。

5　训练成果及测试

5.1 训练成果

输入模拟分布函数ρ图，通过神经网络模型得到预测图，与模拟势能ϕ真值图比较并作差，训练成果由图10-13所示。

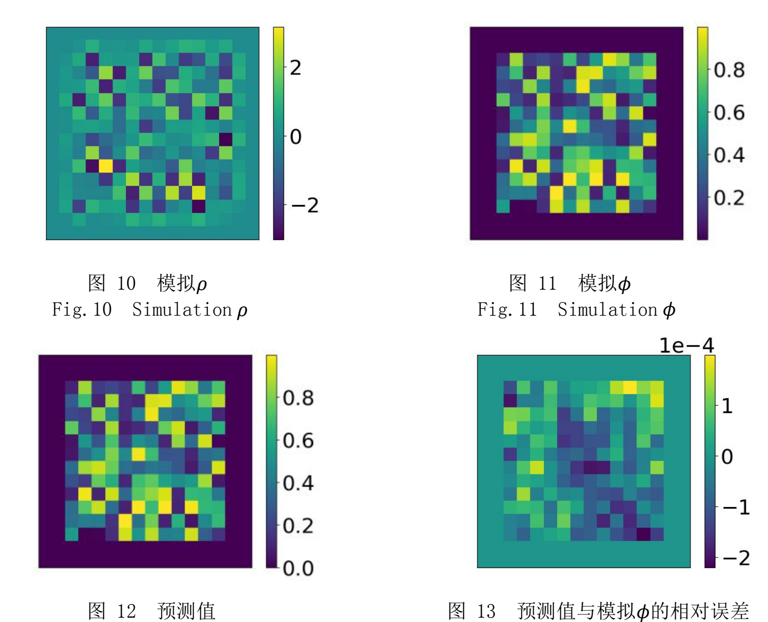


图 10 模拟ρ
Fig.10 Simulation ρ

图 11 模拟ϕ
Fig.11 Simulation ϕ

图 12 预测值
Fig.12 Prediction

图 13 预测值与模拟ϕ的相对误差
Fig.13 Relative error between prediction and simulation ϕ

可以看到，深度神经网络模型的预测图和模拟势能ϕ真值图肉眼几乎看不出区别，图像特征基本吻合，它们的相对误差最大为2×10^{-4}，精度达到Poisson方程的求解目标。

5.2 测试结果

使用一万对新生成数据用于测试，最大误差的数量分布如图14所示，以及统计得到不同最大误差精度下的准确率如表1所示。

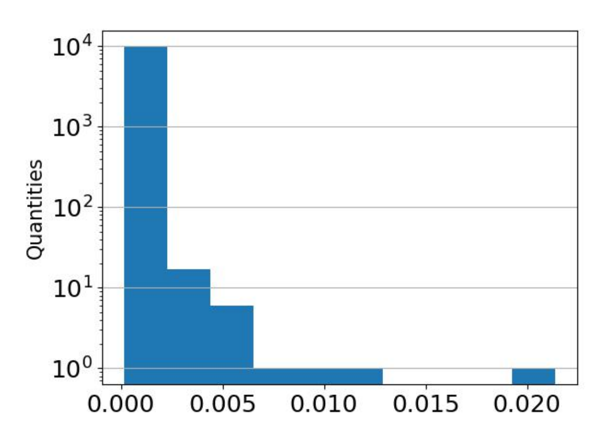


图 14 10000组数据测试结果。横坐标为最大误差，纵坐标为数量
Fig.14 Test results for 10000 sets of data. Horizontal coordinates are maximum error, vertical coordinates are quantities

表 1 不同精度下的准确率
Tab.2 Accuracy at diferent precision

Maximum error	Accuracy
<0.0005	94%
<0.001	99%
<0.005	99.9%

5.3 不同网格化数据测试

为了验证模型的普适性，将用16×16数据训练的神经网络模型用于渐增网格化尺寸($16\times16-1024\times1024$)的图像进行测试，如图15-16所示。

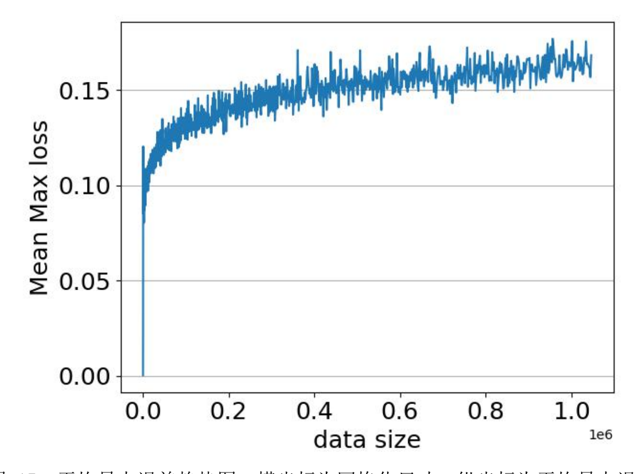


图 15 平均最大误差趋势图。横坐标为网格化尺寸，纵坐标为平均最大误差

Fig.15 Maximum error trend graph. The horizontal coordinate is the mesh size and the vertical coordinate is the average maximum error

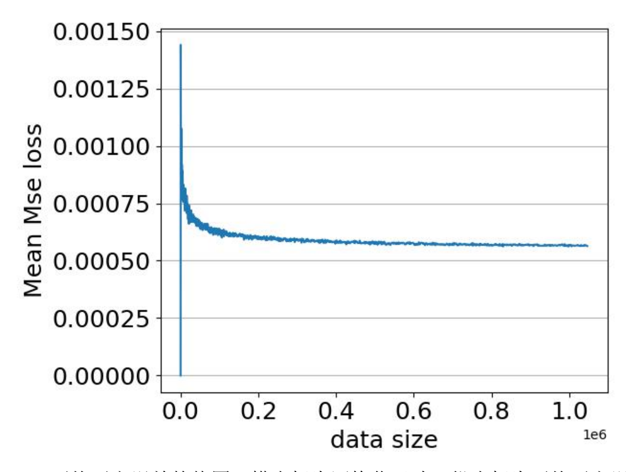
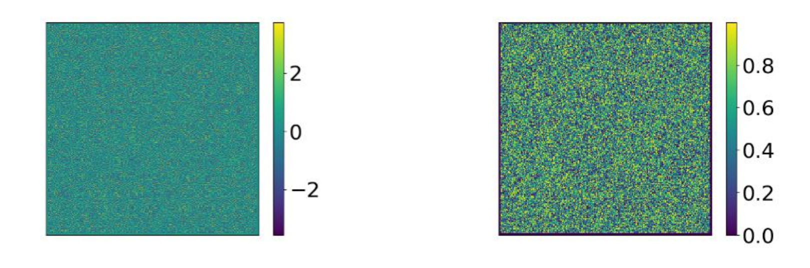


图 16 平均平方误差趋势图。横坐标为网格化尺寸，纵坐标为平均平方误差

Fig.16 Mean squared error trend graph. The horizontal coordinate is the mesh size and the vertical coordinate is the mean squared error

可以看出，最大误差趋势渐平，不会随网格化尺寸增大而线性增大。从平均平方误差图更能看出，随着网格化尺寸增大，平均以后的平方误差反而在减小，当网格化尺寸足够大以后，误差基本不变。取其中两个($256\times256,1024\times1024$)如图17-24示意。



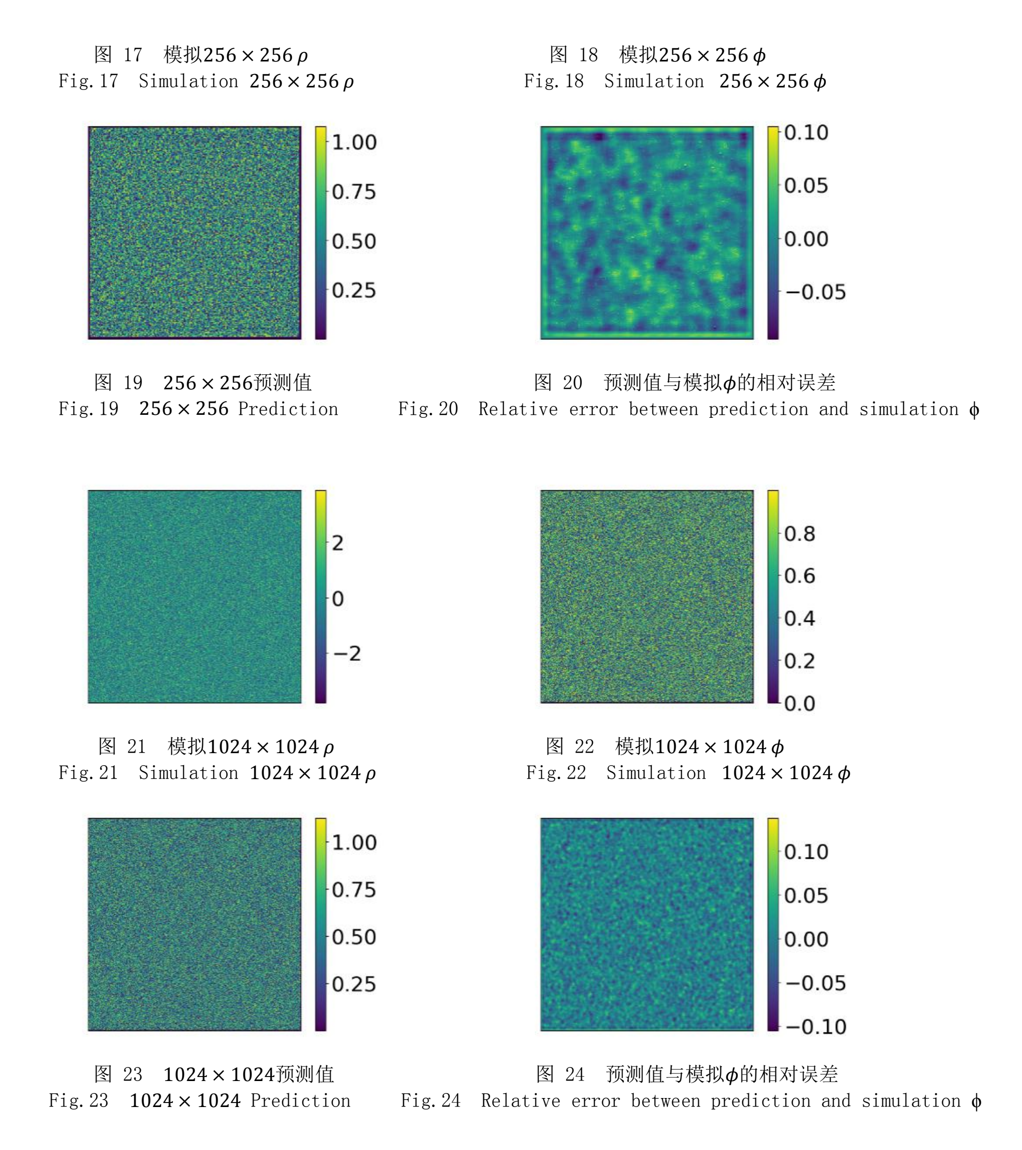


图 17 模拟$256\times256\,\rho$
Fig.17 Simulation $256\times256\,\rho$

图 18 模拟$256\times256\,\phi$
Fig.18 Simulation $256\times256\,\phi$

图 19 256×256预测值
Fig.19 256×256 Prediction

图 20 预测值与模拟ϕ的相对误差
Fig.20 Relative error between prediction and simulation ϕ

图 21 模拟$1024\times1024\,\rho$
Fig.21 Simulation $1024\times1024\,\rho$

图 22 模拟$1024\times1024\,\phi$
Fig.22 Simulation $1024\times1024\,\phi$

图 23 1024×1024预测值
Fig.23 1024×1024 Prediction

图 24 预测值与模拟ϕ的相对误差
Fig.24 Relative error between prediction and simulation ϕ

可以看到，用16×16尺寸数据训练的网络，在更大尺度上依然具有普适性。经过各种尺度图像的对比分析，它的最大误差变大的原因是更大尺寸的数据包含的局部特殊模式更多。而这些特殊模式，就是在测试集数据测试时出现的那些误差较大的点。但总体可以说明，小尺寸数据训练的模型，可以用于大尺寸数据使用。这意味着模型训练的时间成本可以非常低。

5.4 等时势问题测试

我们将训练好的模型用于一个实际的物理问题，等时势问题。直接按公式（11-12）生成等时势问题的ϕ和ρ图像。

$$\rho(r) = M\left[\frac{3(b+a)a^2 - r^2(b+3a)}{4\pi(b+a)^3a^3}\right]\#(11)$$

$$\phi(r) = -\frac{GM}{b+\sqrt{b^2+r^2}}\#(12)$$

将ρ输入到训练好的深度神经网络模型中进行测试，得到预测值，将预测值与模拟ϕ作差对比，如图25-28所示。

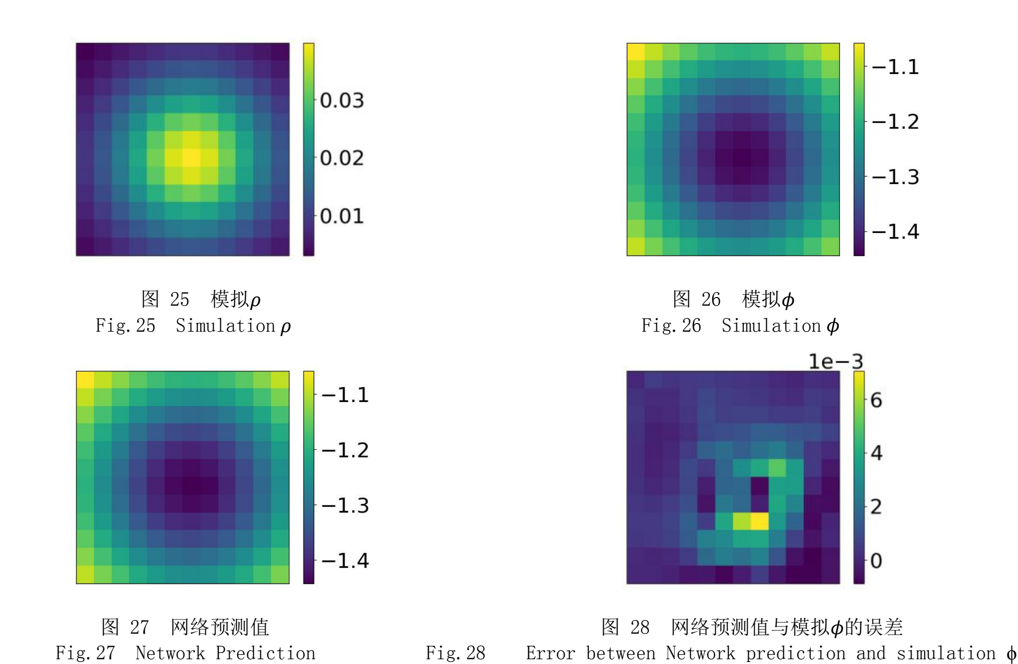


图 25 模拟ρ
Fig.25 Simulation ρ

图 26 模拟ϕ
Fig.26 Simulation ϕ

图 27 网络预测值
Fig.27 Network Prediction

图 28 网络预测值与模拟ϕ的误差
Fig.28 Error between Network prediction and simulation ϕ

可以看到，在实际的物理问题中，深度神经网络模型依然可以表现良好。

同样的数据，我们将之用于快速傅里叶变换方法求解，以做精度对比。如图29-30所示。

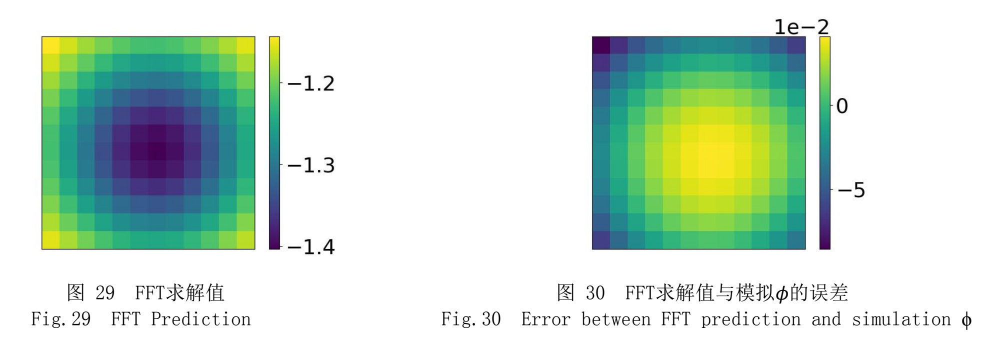


图 29 FFT求解值
Fig.29 FFT Prediction

图 30 FFT求解值与模拟ϕ的误差
Fig.30 Error between FFT prediction and simulation ϕ

图28和图30对比可以看出，在同样采样率的情况下，深度神经网络模型的精度高于快速傅里叶变换方法，满足精度要求。

5.5 与快速傅里叶变换方法及有限差分法的计算时间对比

生成网格化尺寸逐渐增长的数据，分别对比测试训练好的深度神经网络模型和快速傅里叶变换方法以及有限差分法的计算时间。测试结果如图31-32所示。

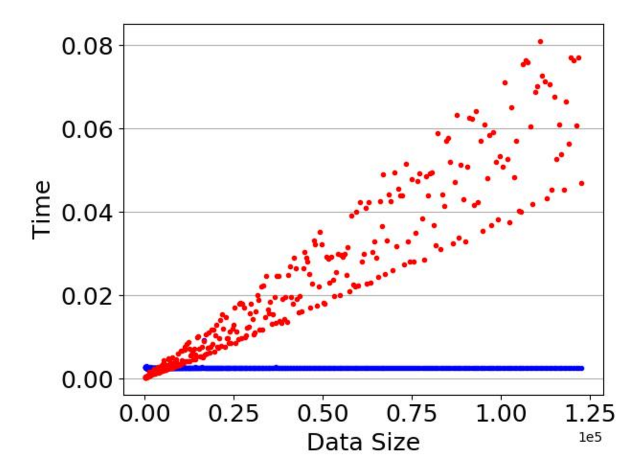


图 31 神经网络模型和快速傅里叶变换方法计算时间对比。横坐标为网格化尺寸，纵坐标为计算时间，单位(s)。蓝色代表神经网络模型，红色代表快速傅里叶变换方法。

Fig. 31 Comparison of neural network model and fast Fourier transform computation times. The horizontal coordinates are the mesh values and the vertical coordinates is the computation time in unit(s). Blue represents the neural network model and red represents the fast Fourier transform.

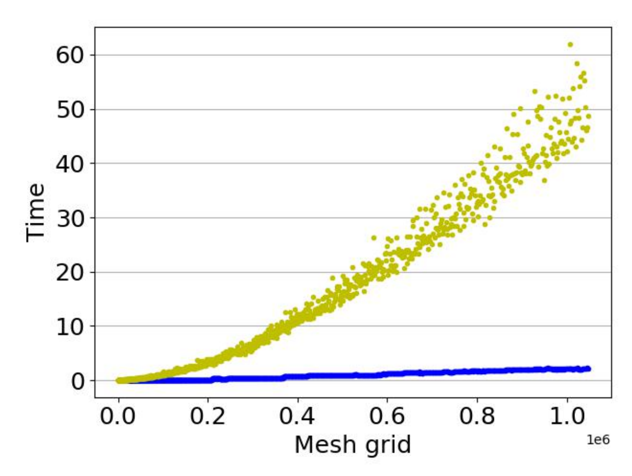


图 32 神经网络模型和有限差分法计算时间对比。横坐标为网格化尺寸，纵坐标为计算时间，单位(s)。蓝色代表神经网络模型，黄色代表有限差分法

Fig.32 Comparison between the neural network model and the finite difference method computation times. The horizontal coordinates are the mesh values and the vertical coordinates are the computation time in unit(s). Blue represents the neural network model, yellow represents the finite difference method

实验结果可以看出，深度神经网络方法求解Poisson方程，随着数据网格化尺寸增大，计算速度快于快速傅里叶变换方法以及有限差分法。这归功于两点，一是深度神经网络可以利用GPU进行快速计算，更重要的一点是，从实验结果图中可以看出深度神经网络的计算时间趋势为线性增长的，而快速傅里叶变换方法为$Nlog(N)$倍增长，有限差分法更是N^2倍增长。这个结论与前文卷积神经网络计算时间复杂度理论分析相互印证。

5.6 失败神经网络模型结构概述

这里描述一些试验结果差的网络结构，数据生成方式和上述相同。第一种是纯卷积网络结构，通道数为：$1-128-256-512-256-128$，最后平均平方误差收敛到0.0003，误差较大。在第一种网络结构的中间每个卷积层后插入归一化层，效果变差，平均平方误差只能收敛到0.14。加入一种网络基础模块，对每次输入进行一次卷积，再将卷积后的输出与输入叠加作为下一层的输入，下一层再通过卷积将维度变回初始输入的维度，效果也不如纯卷积网络，平均平方误差只能收敛到0.005。尝试在纯卷积网络尾端加入两层全连接层，不仅参数量骤增，平均平方误差也仅能达到0.01。再尝试将全连接层加到前端以及后端，平均平方误差能到0.0008，

但参数量很大，训练很慢，且效果还是不如简单的纯卷积网络。尝试将纯卷积网络中间两层的步长改为2，试图提取更精简的特征，并在随后后端的两层将卷积改为反卷积进行上采样，平均平方误差这次降到了10^{-6}，比纯卷积网络效果更好，但依然达不到精度要求。采用一次卷积一次反卷积迭代的网络结构平均平方误差只能到0.3。在纯卷积网络结构上，加入模仿ResNet的网络基础模块，对某次输入进行一次卷积，然后将输入和输出相加作为下一层的输入，一次卷积，一次残差模块，依次6次，平均平方误差能达到10^{-5}。基于以上网络结构的试验，最后综合各种结构的优劣势，设计了最终的深度神经网络模型结构。

5.7 有趣实验现象

我们所给的数据都是设置了Dirichlet边界条件的数据，尝试了一下不设置边界条件会发生什么。用神经网络模型对不设置边界的16×16数据进行训练，训练过程和结果的精度与给定边界条件时的情况相似。在同网格化尺寸下训练结果的相对误差精度也可以达到10^{-4}，但将训练的神经网络模型用于更大网格化尺寸的数据上时，就发生了有趣的现象，它只能预测到边缘16宽度的范围，中间部分几乎为零，如图33-40所示。

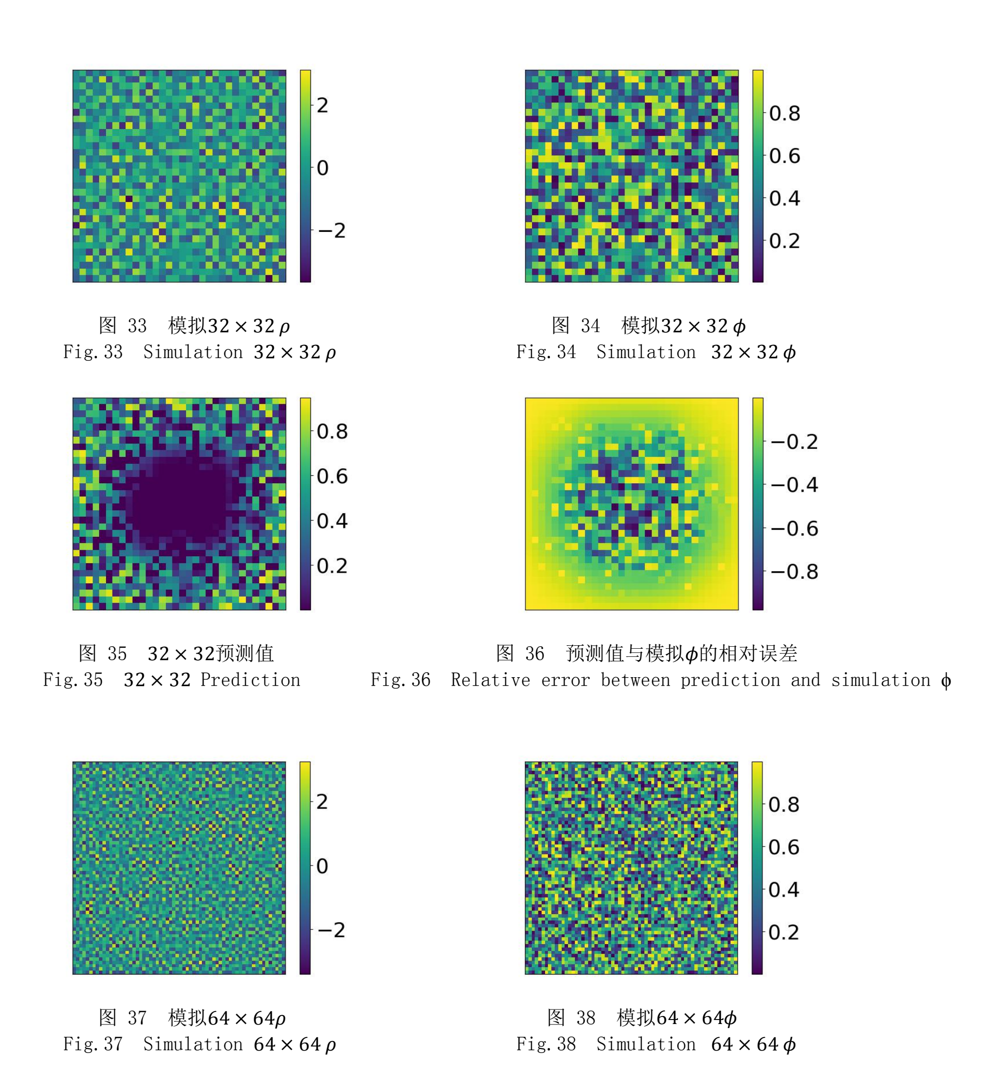


图 33 模拟$32\times32\,\rho$
Fig.33 Simulation $32\times32\,\rho$

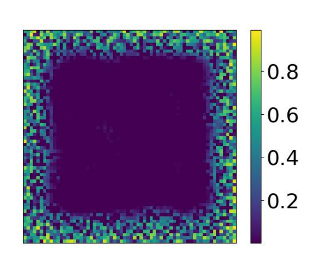
图 34 模拟$32\times32\,\phi$
Fig.34 Simulation $32\times32\,\phi$

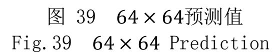
图 35 32×32预测值
Fig.35 32×32 Prediction

图 36 预测值与模拟ϕ的相对误差
Fig.36 Relative error between prediction and simulation ϕ

图 37 模拟$64\times64\rho$
Fig.37 Simulation $64\times64\,\rho$

图 38 模拟$64\times64\phi$
Fig.38 Simulation $64\times64\,\phi$




图 39　64 × 64预测值
Fig.39　64 × 64 Prediction

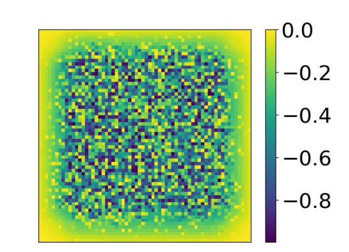


图 40　预测值与模拟ϕ的相对误差
Fig.40　Relative error between prediction and simulation ϕ

结合神经网络模型具有不同网格化数据的普适性来分析，这样的现象可能是因为训练好的神经网络模型是局部迭代操作，同时又是从边界开始向内求解Poisson方程造成的。

5.8 实验总结

综合以上实验结果，可以得出：深度神经网络模型具有快速求解PM-Tree方法中势能（解Poisson方程）的能力；精度不低于快速傅里叶变换法；速度优于快速傅里叶变换法和有限差分法，特别是随着无碰撞引力N体数值模拟的粒子数规模增加，深度神经网络模型的速度优势更加明显；在未经训练的更大网格化尺寸数据上也能使用，具有可扩展性。所以深度学习的方法在加速无碰撞引力N体数值模拟中是可行的。

6　结论

本文在神经网络解各类方程的研究和几种无碰撞引力N体数值模拟方法的基础上，提出了用一个辅以残差局部结构的Encoder-Decoder深度神经网络模型，求解PM-Tree方法中的势能（解Poisson方程）代替常用的FFT求解。验证了深度神经网络模型可以快速求解Poisson方程；理论上深度神经网络的计算时间复杂度为$O(N)$；验证了深度神经网络模型计算耗时低于快速傅里叶变换方法以及有限差分法；验证了在相同离散化采样率的情况下深度神经网络模型求解精度不低于快速傅里叶变换方法；验证了深度神经网络具有可扩展性，在小网格化尺寸数据集上训练的模型可以应用到大网格化尺寸数据集上。并且深度神经网络模型可以更好地利用GPU计算资源进行并行处理，在粒子数规模较大的情况下，速度优势更加明显。综合以上验证结果，我们提出使用深度神经网络代替FFT方法求解PM-Tree方法中的势能（解Poisson方程），可以加速PM-Tree方法中主要的耗时步骤，即可以加速无碰撞引力N体数值模拟，验证了深度学习方法在无碰撞引力N体数值模拟中的可行性。为将来实际应用大规模无碰撞引力N体数值模拟打下基础。

7　未来与展望

未来工作主要有三个方面。第一是拓展解Poisson方程由二维转到三维（只用简单将现有神经网络模型中的二维卷积改为三维卷积即可）。第二，加入迭代求解过程，实现时变的模拟。第三是将模型嵌入到无碰撞引力N体数值模拟的PM-Tree方法中实际应用。

8　致谢

感谢刘冬明、翟顺诚在文章工作中提出的意见；感谢顾建宇对文章部分内容的修正。

Feasibility study of collisionless gravitational N-body

numerical simulation based on deep learning

Zhao Zicheng[1,2], Long Qian[1], Dong Xiaobo[1], Meng Runyu[1,2], Zhong Shiyan[1], Chen Junyi[1], Xiang Zikun[3]

(1.Yunnan Observatories, Chinese Academy of Sciences, Yunnan 652016 , China, Email: zhaozicheng@ynao.ac.cn; 2.University of Chinese Academy of Sciences, Beijing 100049, China; 3.University of Science and Technology of China, Hefei 230026, China)

Abstract: In this paper, a Deep Neural Network is proposed to replace the Fast Fourier Transform method to solve the potential energy in the PM-tree method of the collisionless gravitational N-body numerical simulation, so as to improve the efficiency of the PM-tree method and verify the feasibility of the deep learning method to accelerate the collisionless gravitational N-body numerical simulation. Collisionless gravitational N-body numerical simulations are important for the study of galaxies, dark matter halos, and the formation and evolution of the large-scale structure of the universe. The traditional collisionless gravitational N-body numerical simulation method is very time-consuming for large-scale problems, among which the main time-consuming part of the commonly used PM-tree method is solving the potential energy (solving Poisson equation). In this paper, we propose to use deep neural networks instead of traditional methods to accelerate the solving of Poisson equation, adjust and train and test the model structure of deep neural network for many times, and finally select the overall structure of Encoder-Decoder supplemented with the local structure of residual network. We verify that the computational time complexity of deep neural network to solve Poisson equation is O(N). Tested on the same data, the deep neural network is faster than the fast Fourier transform method solution and the finite difference method solution. At the same sampling rate, the accuracy of deep neural network is better than that of the fast Fourier transform method. And it is extensible. Therefore, in the collisionless gravitational N-body numerical simulation, the Deep Neural Network can improve the velocity of solving the potential energy in the PM-Tree method, so as to effectively improve the overall simulation speed.

Key words: Poisson equation; Gravitational N-body Numerical simulation; Deep Neural Network;

参考文献:

[1] W. Dehnen and J. I. Read. N-body simulations of gravitational dynamics. *The European Physical Journal Plus*, 126(5), May 2011.

[2] Springel, Volker; Pakmor, Rüdiger; Zier, Oliver; Reinecke, Martin. Simulating cosmic structure formation with the GADGET-4 code. arXiv preprint arXiv:2010.03567,2020.

[3] Sverre J Aarseth. *Gravitational N-body simulations: tools and algorithms*. Cambridge University Press, 2003.

[4] Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324(6096):446–449, 1986.

[5] Andrew W Appel. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, 6(1):85–103, 1985.

[6] James W Wadsley, Joachim Stadel, and Thomas Quinn. Gasoline: a flexible, parallel implementation of treesph. *New astronomy*, 9(2):137–158, 2004.

[7] M Gritschneder, T Naab, A Burkert, S Walch, F Heitsch, and M Wetzstein. ivine–ionization in the parallel tree/sph code vine: first results on the observed age-spread around o-stars. Monthly Notices of the Royal Astronomical Society, 393(1):21–31, 2009.

[8] JW Eastwood and RW Hockney. Computer simulation using particles. New York: Mc GrawHill,1981.

[9] Volker Springel, Simon DM White, Adrian Jenkins, Carlos S Frenk, Naoki Yoshida, Liang Gao, Julio Navarro, Robert Thacker, Darren Croton, John Helly, et al. Simulations of the formation,evolution and clustering of galaxies and quasars. nature, 435(7042):629– 636, 2005.

[10] Paul Bode and Jeremiah P Ostriker. Tree particle-mesh: An adaptive, efficient, and parallel code for collisionless cosmological simulation. The Astrophysical Journal Supplement Series, 145(1):1, 2003.

[11] Volker Springel. The cosmological simulation code gadget-2. Monthly notices of the royal astronomical society, 364(4):1105– 1134, 2005.

[12] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. Journal of computational physics, 73(2):325–348, 1987.

[13] Jürgen Schmidhuber. Deep learning in neural networks: An overview. Neural networks, 61:85–117, 2015.

[14] Yoshua Bengio. Learning deep architectures for AI. Now Publishers Inc, 2009.

[15] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989.

[16] Donald F Specht et al. A general regression neural network. IEEE transactions on neural networks, 2(6):568–576, 1991.

[17] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artifficial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks, 9(5):987–1000, 1998.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classifification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[19] Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pages 18–26, 2015.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. nature, 521(7553):436–444, 2015.

[21] Jiequn Han, Arnulf Jentzen, and E Weinan. Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning. arXiv preprint arXiv:1707.02568, pages 1–13, 2017.

[22] J Binney and S Tremaine. Galactic dynamics, ed. Binney, J. & Tremaine, S, 1987.

[23] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 481–490, 2016.

[24] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. Journal of Computational Physics, 366:415–447, 2018.

[25] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flflow fifields using convolutional neural networks. Computational Mechanics, 64(2):525–545, 2019.

[26] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artifificial neural networks. arXiv preprint arXiv:1707.03351, 2017.

[27] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. Inverse Problems, 33(12):124007, 2017.

[28] Wei Tang, Tao Shan, Xunwang Dang, Maokun Li, Fan Yang, Shenheng Xu, and Ji Wu. Study on a poisson's equation solver based on deep learning technique. In 2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS), pages 1–3. IEEE, 2017.

[29] E Weinan and Bing Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 6(1):1–12, 2018.

[30] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. arXiv preprint arXiv:2005.03180, 2020.

[31] Jonathan D Smith, Kamyar Azizzadenesheli, and Zachary E Ross. Eikonet: Solving the eikonal equation with deep neural networks. arXiv preprint arXiv:2004.00361, 2020.

[32] Michele Trenti and Piet Hut. N-body simulations (gravitational). Scholarpedia, 3(5):3930, 2008.

[33] Bagla and Jasjeet S. TreePM: A code for cosmological N-body simulations. Journal of Astrophysics and Astronomy, Volume 23, Issue 3:185-196, 2002.

[34] 张文生. 科学计算中的偏微分方程有限差分法[M]. 高等教育出版社,2006:115-137,172-185.
Zhang Wensheng. Finite Difference Methods for Partial Differential Equations in Science Computation[M]. Higher Education Press, 2006:115-137,172-185.

[35] Roland C Le Bail. Use of fast fourier transforms for solving partial differential equations in physics. Journal of Computational Physics, Volume 9, Issue 3:440-465, 1972.

[36] Shamil Chollampatt and Hwee Tou Ng. A multilayer convolutional encoder-decoder neural network for grammatical error correction. arXiv preprint arXiv:1801.08831, 2018.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[38] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5353–5360, 2015.

[39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.